

EECS 370

Classifying Cache Misses



Announcements

- Lab 10 due Wed
 - Lab 11 meets Fr 11/17 and **M 11/27** (after break)
- P4 out
 - Due Thu 11/30
 - Check out simulator on website (not same as project... doesn't cache instrs)
- HW 4 out
 - Due Mon 12/4

The screenshot displays a simulator interface with three main panels. The left panel, titled 'Instructions ?', contains a 'Load Instructions' dropdown, 'Verify', 'Step >', and '>>' buttons, and a large empty text area. The middle panel, titled 'Data Cache ?', shows a 4x4 grid of cache sets (Set 0 to Set 3). Each set contains four 'INV' (Invalid) blocks. Below the grid, 'Cache Configuration ?' is set to 'Custom'. It also shows '# Hits' as 0 and '# Misses' as 0. At the bottom, 'Block Size (Words)' is 2, '# Sets' is 4, and '# Blocks/Set' is 4. The right panel, titled 'Memory (64 W) ?', shows a vertical stack of 64 memory blocks, indexed 0 to 5 on the right. The blocks are color-coded: green for indices 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63.

What about cache for instructions

- We've been focusing on caching loads and stores (i.e. data)
- Instructions should be cached as well
- We have two choices:
 1. Treat instruction fetches as normal data and allocate cache lines when fetched
 2. Create a second cache (called the **instruction cache** or **ICache**) which caches instructions only
 - More common in practice

How do you know which cache to use?

What are advantages of a separate ICache?

Integrating Caches into Pipeline

- How are caches integrated into a pipelined implementation?
 - Replace instruction memory with Icache
 - Replace data memory with Dcache
- Issues:
 - Memory accesses now have variable latency
 - Both caches may miss at the same time

Agenda

- **Motivation**
- Example
- How to optimize cache design
- Practice Problem 1
- Practice Problem 2
- Practice Problem 3
- Practice Problem 4

Improving our Caches

- If our cache is getting a lot of misses, how do we improve it?
 - Depends on why the misses occurring
 - Is the cache too small? Is the associativity too restrictive? Something else?
- A decent first step is to **classify** the types of missing we are observing

Classifying Cache Misses

- Cache misses happen for 3* reasons
 - The 3C's of Cache misses:
- **Compulsory miss**
 - We've never accessed this data before
- **Capacity miss**
 - Cache is not large enough to hold all the data
 - May have been avoided if we used a bigger cache
- **Conflict miss**
 - Cache is large enough to hold data, but was replaced due to overly restrictive associativity
 - May have been avoided if we used a higher-associative cache

**On multi-core systems, there's a 4th C – take EECS 470/570 to learn more*

Classifying Cache Misses

- Scenario: run given program on system with N-way cache of size M
 - Identify each miss
- We can classify each miss in a program by simulating on 3 different caches
 - If miss still occurs in cache where size \geq memory size: **compulsory miss**
 - Else, if miss occurs in fully associative cache of size M: **capacity miss**
 - Else, if miss occurs in N-way cache of size M (original cache): **conflict miss**

Agenda

- Motivation
- **Example**
- How to optimize cache design
- Practice Problem 1
- Practice Problem 2
- Practice Problem 3
- Practice Problem 4

3C's Sample Problem

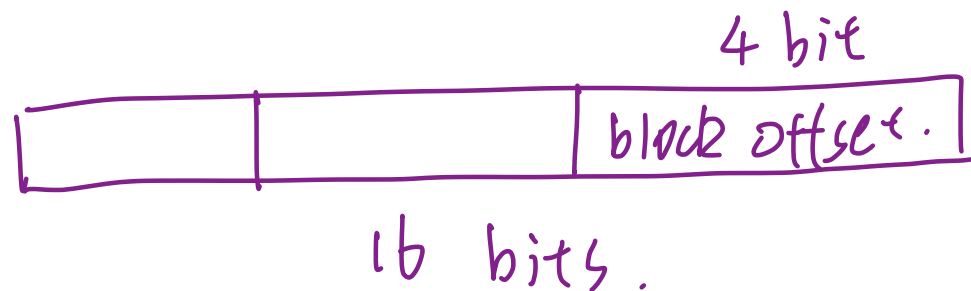
4 block

2 sets

Consider a cache with the following configuration: write-allocate, total size is 64 bytes, block size is 16 bytes, and 2-way associative. The memory address size is 16 bits and byte-addressable. The replacement policy is LRU. The cache is empty at the start.

2⁴

For the following memory accesses, indicate whether the reference is a hit or miss, and the type of a miss (compulsory, conflict, capacity)



64 bytes. 64

4 block.

infinity
Q

3 C's Practice Problem – 3 C's

64 bytes total, 16 byte blocks, 2-way, 2 sets

2 set. 1 bit.

Address	Infinite	FA	SA	3Cs
0x00	M	M	M	Compulsory
0x14	M	M	M	Compulsory
0x27	M	M	M	Compulsory
0x08	H	H	H	
0x38	M	M	M	Compulsory
0x4A	M	M	M	Compulsory
0x18	H	M	H	
0x27	H	M	M	Capacity
0x0F	H	M	M	Capacity
0x40	H	H	M	Conflict.

FA
2
x 4
2 1
2 0
SA
2 4
2 0
Set 0
1
3 set 1

3 C's Practice Problem – 3 C's

64 bytes total, 16 byte blocks, 2-way, 2 sets

since I only care about "hit" or "miss" so I don't care about block offset here.

16 bytes → 4 bits

Address	Infinite	FA	SA	3Cs
0x00	Miss	Miss	Miss	Comp
0x14	Miss	Miss	Miss	Comp
0x27	Miss	Miss	Miss	Comp
0x08	hit	hit	hit	—
0x38	Miss	Miss	Miss	Comp
0x4A	Miss	Miss	Miss	Comp
0x18	hit	Miss	hit	—
0x27	hit	Miss * <small>make cache larger</small>	Miss	Compairty
0x0F	hit	Miss *	Miss	Compairty
0x40	hit	hit	Miss	Conflict

FA

0	2
1	4
2	1
3	0

SA

0	4 Set 0
2	0

1	Set 1
3	

3 C's Practice Problem – 3 C's

Poll: How many blocks will be in a 64 byte FA cache?

Address	Infinite	FA	SA	3Cs
0x00	M			
0x14	M			
0x27	M			
0x08	H			
0x38	M			
0x4A	M			
0x18	H			
0x27	H			
0x0F	H			
0x40	H			

3 C's Practice Problem – 3 C's

64 bytes total, 16 byte blocks, 2-way, 2 sets

Address	Infinite	FA	SA	3Cs
0x00	M	M		
0x14	M	M		
0x27	M	M		
0x08	H	H		
0x38	M	M		
0x4A	M	M		
0x18	H	M		
0x27	H	M		
0x0F	H	M		
0x40	H	H		

3 C's Practice Problem – 3 C's

64 bytes total, 16 byte blocks, 2-way, 2 sets

Address	Infinite	FA	SA	3Cs
0x00	M	M	M	Compulsory
0x14	M	M	M	Compulsory
0x27	M	M	M	Compulsory
0x08	H	H	H	---
0x38	M	M	M	Compulsory
0x4A	M	M	M	Compulsory
0x18	H	M	H	---
0x27	H	M	M	Capacity
0x0F	H	M	M	Capacity
0x40	H	H	M	Conflict

Agenda

- Motivation
- Example
- **How to optimize cache design**
- Practice Problem 1
- Practice Problem 2
- Practice Problem 3
- Practice Problem 4

How to reduce cache misses

- **Compulsory miss**

- Reduce by **increasing cache block size**
 - Reduces total number of blocks for given cache size ☹️
- Or by using prefetching (guess we'll need data based on previous memory patterns - discussed more in EECS 470)

- **Capacity miss**

- Reduce by **building a bigger cache**
 - Increase access latency ☹️

- **Conflict miss**

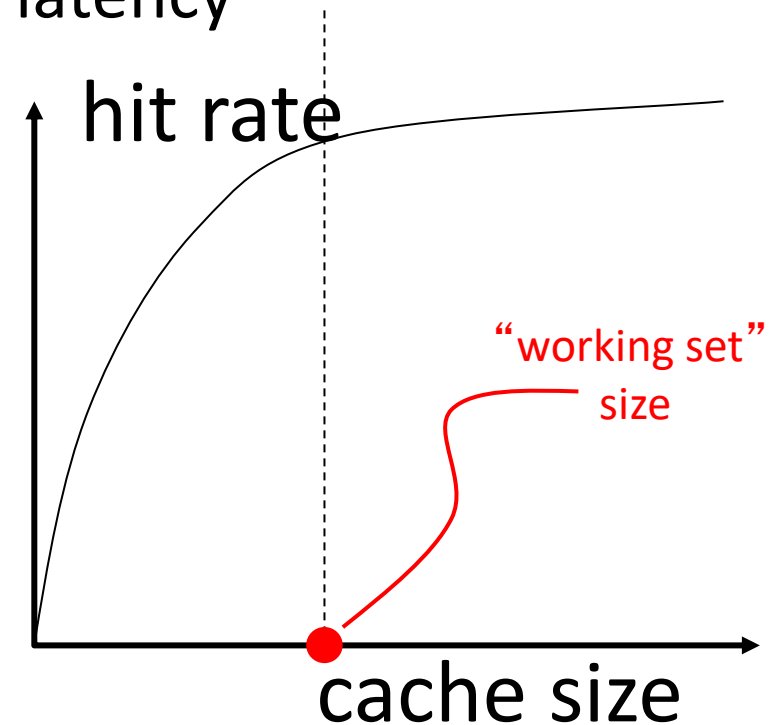
- Reduce by **increasing associativity**
 - Increase access latency / overheads ☹️

Cache Performance

- How does changing these parameters affect performance?
 - Cache size
 - Block size
 - Associativity

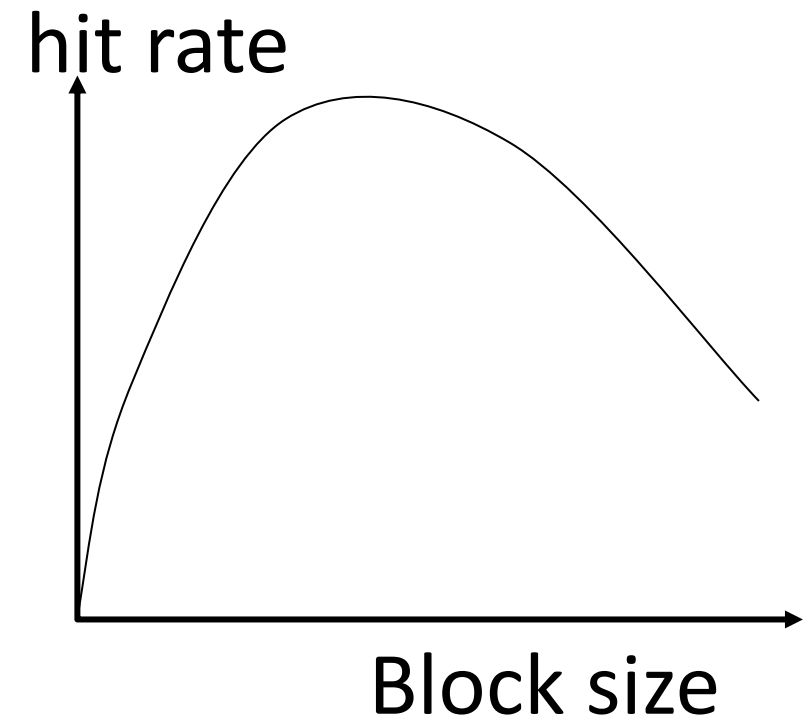
Cache Size

- Cache size in the total data (not including tag) capacity
 - bigger can exploit temporal locality better
 - not ALWAYS better
- Too large a cache adversely affects hit & miss latency
 - smaller is faster => bigger is slower
 - access time may degrade critical path
- Too small a cache
 - doesn't exploit temporal locality well
 - useful data replaced often
- **Working set**: the whole set of data executing application references
 - **Within a time interval**



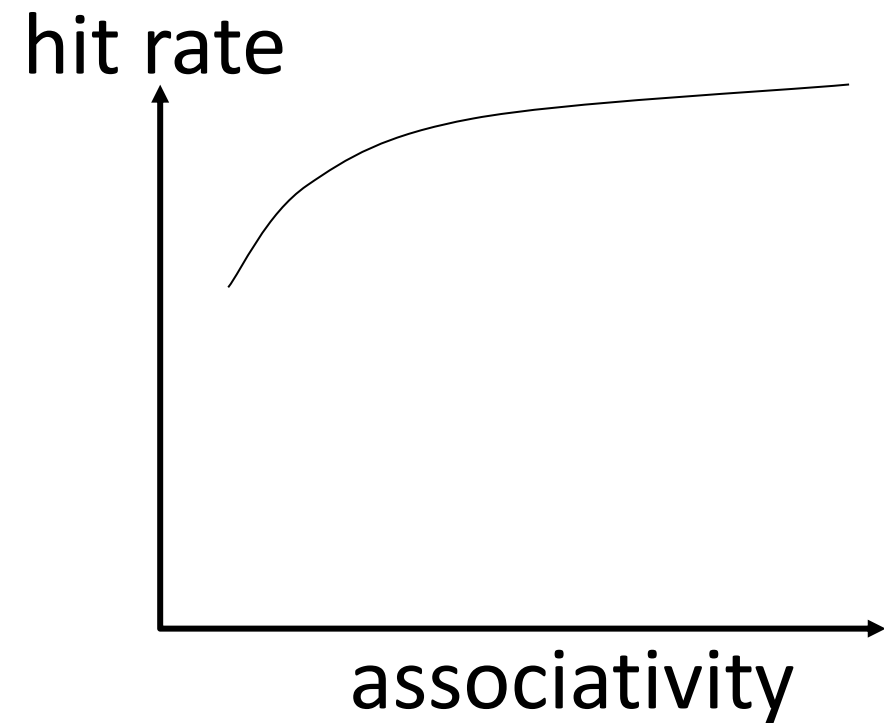
Block size

- Block size is the data that is associated with an address tag
 - Sub-blocking: A block divided into multiple pieces (each with V bit)
 - Can improve “write” performance
 - Take 470 to learn more
- Too small blocks
 - don’t exploit spatial locality well
 - have larger tag overhead
- Too large blocks
 - too few total # of blocks
 - likely-useless data transferred
 - Extra bandwidth/energy consumed



Associativity

- How many blocks can map to the same index (or set)?
- Larger associativity
 - lower miss rate, less variation among programs
 - diminishing returns
- Smaller associativity
 - lower cost
 - faster hit time
 - Especially important for L1 caches



Agenda

- Motivation
- Example
- How to optimize cache design
- **Practice Problem 1**
- Practice Problem 2
- Practice Problem 3
- Practice Problem 4

Poll: What is the CPI increase over base (1) due to hazards?

Practice Problem 1: CPI with caches

The *blaster* application run on the LC2k with full data forwarding and all branches predicted not-taken has the following instruction frequencies:

45% R-type 20% Branches 15% Loads 20% Stores

In *blaster*, 40% of branches are taken and 50% of LWs are followed by an immediate use.

The I-cache has a miss rate of 3% and the D-cache has a miss rate of 6% (no overlapping of misses). On a miss, the main memory is accessed and has a latency of 100 ns. The clock frequency is 500 MHz.

$$1 + 0.2 \times 0.4 \times 3 + 0.15 \times 0.5 \times 1 + 1 \times 0.03 \times 50 + 0.35 \times 0.06 \times 50$$

Handwritten calculations:

$$\frac{1}{500 \times 10^6} = \frac{1}{0.5 \times 10^9} = 2 \text{ ns.}$$
$$100 \text{ ns} \div 2 \text{ ns} = 50 \text{ cycle}$$

Problem 1 Solution

The *blaster* application run on the LC2k with full data forwarding and all branches predicted not-taken has the following instruction frequencies:

45% R-type 20% Branches 15% Loads 20% Stores

In *blaster*, 40% of branches are taken and 50% of LWs are followed by an immediate use.

The I-cache has a miss rate of 3% and the D-cache has a miss rate of 6% (no overlapping of misses). On a miss, the main memory is accessed and has a latency of 100 ns. The clock frequency is 500 MHz.

What is the CPI of *blaster* on the LC2k?

Stalls per cache miss = 100 ns / 2ns = 50 cycles (500 Mhz → 2ns cycle time)

CPI = 1 + data hazard stalls + control hazard stalls + icache stalls + dcache stalls

CPI = 1 + 0.15*0.50*1 + 0.20*0.40*3 + 1*0.03*50 + 0.35*0.06*50

Agenda

- Motivation
- Example
- How to optimize cache design
- Practice Problem 1
- **Practice Problem 2**
- Practice Problem 3
- Practice Problem 4

Poll: How many bytes are read/written if we have no cache?

Practice Problem 2: Memory Usage

- Say you have the following: 2×10^{12} lw 1×10^{12} sw
 - A program that generates 2 Billion loads and 1 Billion stores, each 4 bytes in size.
 - A cache with a 32-byte block which gets a 95% hit rate on that program.
- How many bytes of memory would be read and written if:
 - We had no cache? $(2+1) \times 10^{12} \times 4 = 12 \times 10^{12} = 1.2 \times 10^{13}$
 - We had a write-through cache with a no-write allocate policy?
 - We had a write-back cache with a write-allocate policy? (Assume 25% of all misses result in a dirty eviction)

For sw : each time will update the memory: $1 \times 10^{12} \times 4 = 4 \times 10^{12}$

For lw : consider, if hit \rightarrow grab from cache

if miss \rightarrow grab from memory : $2 \times 10^{12} \times 32 \times 0.05 = 3.2 \times 10^{12}$

(But not only load 4 bytes, instead, it's 32 bytes).

Poll: How many bytes are read/written if we have no cache?

Practice Problem 2: Memory Usage

- Say you have the following:
 - A program that generates 2 Billion loads and 1 Billion stores, each 4 bytes in size.
 - A cache with a 32-byte block which gets a 95% hit rate on that program.
- How many bytes of memory would be read and written if:
 - We had no cache?
 - We had a write-through cache with a no-write allocate policy?
 - We had a write-back cache with a write-allocate policy? (Assume 25% of all misses result in a dirty eviction)

2 + 4 = 6 Billion

L: load from cache: -
 miss: load from memory

- full and dirty is 1: $2 \times 10^{12} \times 32 \times 0.05 \times 0.25 \times 2 = 1.6$ Billion
- not full or full and dirty is 0: $2 \times 10^{12} \times 32 \times 0.05 \times 0.75 = 2.4$ Billion

SW:

- ① hit: load from cache: -
- ② miss: load from memory
 - full and dirty is 1: $1 \times 10^{12} \times 32 \times 0.05 \times 0.25 \times 2 = 0.8$ Billion
 - not full or full and dirty is 0: $1 \times 10^{12} \times 32 \times 0.05 \times 0.75 = 1.2$ Billion

Practice Problem 2: Memory Usage

- Say you have the following:
 - A program that generates 2 Billion loads and 1 Billion stores, each 4 bytes in size.
 - A cache with a 32-byte block which gets a 95% hit rate on that program.
- Let's start with the no-cache case.
 - All stores go to memory and are 4 bytes each
 - Writes: 1 billion stores * 4 bytes = 4 billion bytes
 - All loads go to memory and are 4 bytes each.
 - Reads: 2 billion loads * 4 bytes = 8 billion bytes
- Write-through with no write-allocate?

Practice Problem 2: Memory Usage

- Say you have the following:
 - A program that generates 2 Billion loads and 1 Billion stores, each 4 bytes in size.
 - A cache with a 32-byte block which gets a 95% hit rate on that program.
- Write-through, no allocate.
 - All stores still go to memory and are still 4 bytes each.
 - Writes: $1 \text{ billion stores} * 4 \text{ bytes} = 4 \text{ billion bytes}$
 - Only loads that miss in the cache go to memory. But they read the full cache block.
 - Reads: $2 \text{ billion loads} * 0.05 * 32 \text{ bytes} = 3.2 \text{ billion bytes}$

Practice Problem 2: Memory Usage

- Say you have the following:
 - A program that generates 2 Billion loads and 1 Billion stores, each 4 bytes in size.
 - A cache with a 32-byte block which gets a 95% hit rate on that program.
- Write-through, no allocate.

Practice Problem 2: Memory Usage

- Say you have the following:
 - A program that generates 2 Billion loads and 1 Billion stores, each 4 bytes in size.
 - A cache with a 32-byte block which gets a 95% hit rate on that program.
- Write-back, write-allocate (25% of all misses result in a dirty eviction)

Practice Problem 2: Memory Usage

- Say you have the following:
 - A program that generates 2 Billion loads and 1 Billion stores, each 4 bytes in size.
 - A cache with a 32-byte block which gets a 95% hit rate on that program.
- Write-back, write-allocate (25% of all misses result in a dirty eviction)
 - *Store* misses result in a cache block being read.
 - Reads: $1 \text{ billion stores} * 0.05 * 32 \text{ bytes} = 1.6 \text{ billion bytes}$
 - *Load* misses result in a cache block being read.
 - Reads: $2 \text{ billion loads} * 0.05 * 32 \text{ bytes} = 3.2 \text{ billion bytes}$
 - So that is 4.8 billion bytes of data read.

Practice Problem 2: Memory Usage

- Say you have the following:
 - A program that generates 2 Billion loads and 1 Billion stores, each 4 bytes in size.
 - A cache with a 32-byte block which gets a 95% hit rate on that program.
- Write-back, write-allocate (25% of all misses result in a dirty eviction)
 - *Store* misses result in dirty eviction 1/4 of the time.
 - Reads: $1 \text{ billion stores} * 0.05 * 32 \text{ bytes} * (.25) = 0.4 \text{ billion bytes}$
 - *Load* misses result in a cache block being read.
 - Reads: $2 \text{ billion loads} * 0.05 * 32 \text{ bytes} * (.25) = 0.8 \text{ billion bytes}$

Agenda

- Motivation
- Example
- How to optimize cache design
- Practice Problem 1
- Practice Problem 2
- **Practice Problem 3**
- Practice Problem 4

Practice Problem 3: CPI w/ Caches 2

$$\frac{1}{2 \times 10^8} = \frac{1}{0.2 \times 10^9} = 5 \text{ ns} \leftarrow 1 \text{ cycles.}$$

- Given a 200 MHz processor with 8KB instruction and data caches and a with memory access latency of 20 cycles. Both caches are 2-way associative. A program running on this processor has a 95% icache hit rate and a 90% dcache hit rate. On average, 30% of the instructions are loads or stores. The CPI of this system, if caches were ideal would be 1.
- Suppose you have 2 options for the next generation processor, which do you pick?
$$CPI = 1 + 0.3 \times 0.1 \times 40 + 1 \times 0.05 \times 40 = 1 + 1.2 + 2 = 4.2$$
 - Option 1:** Double the clock frequency—assume this will increase your memory latency to 40 cycles. Also assume a base CPI of 1 can still be achieved after this change.
$$4.2 \times 5 \text{ ns} \times 0.5 = 10.5 \text{ ns}$$
 - Option 2:** Double the size of your caches, this will increase the instruction cache hit rate to 98% and the data cache hit rate to 95%. Assume the hit latency is still 1 cycle.
$$CPI = 1 + 0.3 \times 0.05 \times 20 + 1 \times 0.02 \times 20 = 1 + 0.3 + 0.4 = 1.7$$
$$1.7 \times 5 \text{ ns} = 8.5 \text{ ns.}$$

Practice Problem 3: CPI w/ Caches 2

- Given a 200 MHz processor with 8KB instruction and data caches and a with memory access latency of 20 cycles. Both caches are 2-way associative. A program running on this processor has a 95% icache hit rate and a 90% dcache hit rate. On average, 30% of the instructions are loads or stores. The CPI of this system, if caches were ideal would be 1.
- Suppose you have 2 options for the next generation processor, which do you pick?
 - **Option 1:** Double the clock frequency—assume this will increase your memory latency to 40 cycles. Also assume a base CPI of 1 can still be achieved after this change.
 - **Option 2:** Double the size of your caches, this will increase the instruction cache hit rate to 98% and the data cache hit rate to 95%. Assume the hit latency is still 1 cycle.

Practice Problem 3: Solution

Option 1: (double clock freq, base cycle time is 5 ns, so new cycle time is 2.5 ns)

$$\text{CPI} = \text{baseCPI} + \text{IcacheStallCPI} + \text{DcacheStallCPI}$$

$$\text{CPI} = 1.0 + 0.05 * 40 + 0.3 * 0.1 * 40 = 4.2$$

$$\underline{\text{Execution time} = 4.2 * \text{Ninstrs} * 2.5\text{ns} = 10.5\text{ns} * \text{Ninstrs}}$$

Option 2 (icache/dcachel miss rates lowered to 2% and 5%)

$$\text{CPI} = \text{baseCPI} + \text{IcacheStallCPI} + \text{DcacheStallCPI}$$

$$\text{CPI} = 1.0 + 0.02 * 20 + 0.3 * 0.05 * 20 = 1.7$$

$$\underline{\text{Execution time} = 1.7 * \text{Ninstrs} * 5\text{ns} = 8.5\text{ns} * \text{Ninstrs}}$$

Therefore, Option 2 is the better choice

Agenda

- Motivation
- Example
- How to optimize cache design
- Practice Problem 1
- Practice Problem 2
- Practice Problem 3
- **Practice Problem 4**

Practice Problem 4: Guess that cache!

Here is the series of address references (in hex) to a cache of size 512 bytes. You are asked to **determine the configuration of the cache**. Assume 12-bit addresses

0011 0001 0000 if blocks were ≥ 32 bytes.
0011 0000 1111

0x310 - Miss
0x30f - Miss block size can't be too large.
0x510 - Miss
0x31f - Hit 0011 0001 0000 block size ≥ 16
0x72d - Miss
0x72f - Hit
0x320 - Miss
0x520 - Miss
0x720 - Miss

block size < 32
block size = 16

Block size: ? 16
Associativity: ?
Number of sets: ?

Practice Problem 4: Guess that cache!

Here is the series of address references (in hex) to a cache of size 512 bytes. You are asked to **determine the configuration of the cache**. Assume 12-bit addresses

11
10
11
11X
12
12
12
12
12

0x310	- Miss	1
0x30f	- Miss	0
0x510	- Miss	1
0x31f	- Hit	1
0x72d	- Miss	2
0x72f	- Hit	2
0x320	- Miss	2
0x520	- Miss	2
0x720	- Miss	2

S I ①

impossible directly map cache.

since 0x510 will replace 0x310.
and 0x31f is impossible to have hit.

② Could it be two-way cache?

每个set里两个way. 1 bit tag. 4bits set index.

72 52
32 72

only if it's two-way.

number of blocks: $2^9 / 2^4 = 2^5$ lines

$\frac{2^5}{2} = 2^4$ sets

Start with directly map cache

Block size: ? 16

Associativity: ?

Number of sets: ?

I should hit
but actually miss.

24 sets

Practice Problem 4: Guess that cache!

Similar to homework! Here is the series of address references (in hex) to a cache of size 512 bytes. You are asked to **determine the configuration of the cache**. Assume 12-bit addresses

0x310 - Miss

0x30f - Miss

0x510 - Miss

0x31f - Hit

0x72d - Miss

0x72f - Hit

0x320 - Miss

0x520 - Miss

0x720 - Miss

Practice Problem 4: Guess that cache!

Similar to homework! Here is the series of address references (in hex) to a cache of size 512 bytes. You are asked to **determine the configuration of the cache**. Assume 12-bit addresses

0x310 - Miss

0x30f - Miss

0x510 - Miss

0x31f - Hit

0x72d - Miss

0x72f - Hit




0x320 - Miss

0x520 - Miss

0x720 - Miss

Practice Problem 4: Guess that cache!

Here is the series of address references (in hex) to a cache of size 512 bytes. You are asked to **determine the configuration of the cache**. Assume 12-bit addresses

0x310 – Miss 
0x30f – Miss 
0x510 – Miss
0x31f – Hit 
0x72d – Miss
0x72f – Hit
0x320 – Miss
0x520 – Miss
0x720 – Miss

Determine block size

First hit must be brought in by another miss








Take closest address: 0x310, so know block size must be at least 16 bytes so 0x31f brought in when 0x310 miss occurs

Now, is the block size larger? Know that 0x30f was a miss, thus 0x310 and 0x30f not in the same block. Thus, block size must be ≤ 16 bytes

Thus Block Size = 16 bytes

Practice Problem 4: Guess that cache!

Here is the series of address references (in hex) to a cache of size 512 bytes. You are asked to **determine the configuration of the cache**. Assume 12-bit addresses

0x310 – Miss 
0x30f – Miss
0x510 – Miss 
0x31f – Hit 
0x72d – Miss
0x72f – Hit 
0x320 – Miss 
0x520 – Miss 
0x720 – Miss 

Determine associativity

Assume direct mapped: 3-bit tag, 5-bit index, 4-bit offset.
If DM, 0x310 and 0x510 would both map to index 17,
Thus 0x31f could not be a hit. So, not direct mapped.

Assume 2-way associative: 4-bit tag, 4-bit index, 4-bit offset
This fixes the green accesses, and allows 0x31f to be a hit.

What about > 2-way associative?

Now we also know that 0x720 is a miss even though 3 accesses earlier 0x72f was a hit, and thus it is in the cache. The intervening 2 accesses must kick it out, 0x320 and 0x520. Both go to set 2. If the associativity was > 2, then 0x720 would be a hit. So, must conclude that cache is 2-way associative.

Lastly, number of sets = $512 / (2 * 16) = 16$

Next time

- Completing the hierarchy: virtual memory